

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE EXAMPLE SYSTEM "CSDL_Example.dtd">
```

HP docket 10008278-1
Appendix

```
<Composite-Service>
```

```
  <Meta-Model Name="Food Delivery" Version="A.01.00">
```

```
    <!-- Case packet data. These are the definitions of variables used by this composite service. -->
```

```
    <Composite-Service-Data>
```

```
      <Data-Item-Declaration Name="RestaurantName" Data-Type="STRING">
        <Value>%RestaurantName</Value>
      </Data-Item-Declaration>
      <Data-Item-Declaration Name="OrderNumber" Data-Type="INTEGER">
        <Value>%OrderNumber</Value>
      </Data-Item-Declaration>
      <Data-Item-Declaration Name="Order" Data-Type="STRING">
        <Value>%Order</Value>
      </Data-Item-Declaration>
      <Data-Item-Declaration Name="CustomerCCN" Data-Type="STRING">
        <Value>%CustomerCCN</Value>
      </Data-Item-Declaration>
      <Data-Item-Declaration Name="DeliveryTime" Data-Type="STRING">
        <Value>%DeliveryTime</Value>
      </Data-Item-Declaration>
      <Data-Item-Declaration Name="DeliveryAddress" Data-Type="STRING">
        <Value>%DeliveryAddress</Value>
      </Data-Item-Declaration>
      <Data-Item-Declaration Name="CustomerName" Data-Type="STRING">
        <Value>%CustomerName</Value>
      </Data-Item-Declaration>
      <Data-Item-Declaration Name="Confirmation" Data-Type="INTEGER">
        <Value>%Confirmation</Value>
      </Data-Item-Declaration>
      <Data-Item-Declaration Name="RestaurantCCN" Data-Type="STRING">
        <Value>%RestaurantCCN</Value>
      </Data-Item-Declaration>
      <Data-Item-Declaration Name="CCExpiry" Data-Type="STRING">
        <Value>%CCExpiry</Value>
      </Data-Item-Declaration>
      <Data-Item-Declaration Name="BillAmount" Data-Type="REAL">
        <Value>%CustomerName</Value>
      </Data-Item-Declaration>
```

```
    <!-- The following two variables are used in Search-Recipe. Search-Recipe is used in E-speak
```

```
      to look up for services. -->
```

```
    <Data-Item-Declaration Name="mycondition" Data-Type="STRING">
      <Value>%mycondition</Value>
    </Data-Item-Declaration>
    <Data-Item-Declaration Name="mypreference" Data-Type="STRING">
      <Value>%mypreference</Value>
    </Data-Item-Declaration>
    <Data-Item-Declaration Name="localhost" Data-Type="STRING">
      <Value>%localhost</Value>
    </Data-Item-Declaration>
```

```
    <!-- Input variables -->
```

```
    <Input>
      <Data-Item Name="CustomerCCN" />
      <Data-Item Name="CCExpiry" />
      <Data-Item Name="Order" />
      <Data-Item Name="DeliveryTime" />
      <Data-Item Name="DeliveryAddress" />
      <Data-Item Name="mycondition" />
      <Data-Item Name="mypreference" />
      <Data-Item Name="localhost" />
    </Input>
```

```
    <!-- Output variables -->
```

```
    <Output>
      <Data-Item Name="Confirmation" />
```

HP docket 10008278-1
Appendix

```

<Data-Item Name="RestaurantName" />
<Data-Item Name="OrderNumber" />
<Data-Item Name="BillAmount" />
</Output>

<!-- Local variables: used only within the composite service -->
<Local>
  <Data-Item Name="RestaurantCCN" />
  <Data-Item Name="CustomerName" />
</Local>

</Composite-Service-Data>

<!-- Service-Flow-Structure describes the service flow and consists of route nodes,
method nodes, and arcs -->
<Service-Flow-Structure>

  <!-- If the user's credit card was confirmed, the service continues to with delivery
      schedule; otherwise the service aborts. The condition below checks the value
      of the Confirmation variable in order to figure out whether the credit card was
      confirmed. -->
  <Route-Node Name="Check Passed" Description="Is CCN confirmed" Type="XOR-SPLIT">
    <Rule>
      <Condition>Confirmation &lt;&gt; 0</Condition>
      <Action>Goto Node Check And Join</Action>
      <Alternative-Action>Goto Node Abort</Alternative-Action>
    </Rule>
  </Route-Node>

  <!-- This is a join node used for joining two separate branches of the service flow, after the
      user's credit card is confirmed and the restaurant is selected. -->
  <Route-Node Name="And Join" Description="Join node" Type="AND-JOIN">
    <Rule>
      <Condition>TRUE</Condition>
      <Action>Goto Node Wheel Delivery</Action>
    </Rule>
  </Route-Node>

  <!-- This service carries out the confirmation of user's credit card. It takes one output, namely
      the credit card number, and returns a confirmation number. If the credit check fails, then
      the return value equals to zero -->
  <Service-Node Name="Check Credit" Description="Confirm credit card number">
    <!-- Search-recipe is used for service lookup in E-speak. It is used in the composite service
        in order to lookup the individual E-speak services that constitute the composite service.
        There are three workflow variables embedded inside the Search-Recipe: localhost,
        mycondition, and mypreference. These variables are replaced by their current values
        by the Gateway during the execution of the composite service. The localhost variable
        contains the URI of the local host machine of the user who is using the composite service.
        The mycondition variable contains the search criteria. The mypreference variable contains
        the sorting criteria in case multiple candidate services are found by the search recipe. All
        those three variables are input to the composite service by the user. This search recipe
        returns the first service that satisfies the given condition and is ranked the first after sorting
        preference is applied. -->
    <Search-Recipe>
      <?xml version="1.0"?>
      <header xmlns="www.e-speak.net/Schema/E-speak.header.xsd">
        <communication>
          <to>es://localhost/WebAccess/FindService</to>
          <from>es://localhost/WAUser1</from>
        </communication>
      </header>
      <esquery xmlns="www.e-speak.net/Schema/E-speak.query.xsd">
        <from>src="es:%localhost"/>
        <result>$serviceinfo</result>
        <where>%mycondition</where>
        <preference>%mypreference</preference>
        <!-- Arbitration is fixed to only one cardinality since we want only one result -->
        <arbitration>
          <operator>first</operator>
          <cardinality>1</cardinality>
        </arbitration>
      </esquery>
    </Search-Recipe>
  </Service-Node>
</Service-Flow-Structure>

```

HP docket 10008278-1
Appendix

```
</arbitration>
</esquery>
</Search-Recipe>
```

```
<!-- The following method confirms the user's credit card. -->
<Method-Node Name="Check_CCN_Node" Description="Confirm CCN">
  <Method-Name>CheckCCN</Method-Name>
  <Method-Input>
    <Value>%CustomerCCN</Value>
  </Method-Input>
  <Method-Output>
    <Var-Mapping FLOW-VAR="Confirmation" />
  </Method-Output>
</Method-Node>
<!-- Example certificate: client's certificate is used -->
<Certificate Type="USER" />
<!-- Skipped the optional Service-Exception-Handling element -->
</Service-Node>
```

```
<!-- This service performs the restaurant selection. It takes two inputs: user order and
     delivery time. It returns the name of the selected restaurant. It consists of only one method
     which performs the restaurant selection. -->
```

```
<Service-Node Name="Restaurant Selection" Description="Select a restaurant">
  <Search-Recipe>
    <!-- Search recipe is the same as that of Check Credit node -->
  </Search-Recipe>
  <Method-Node Name="Select_Restaurant_Node" Description="Select Restaurant">
    <Method-Name>SelectRestaurant</Method-Name>
    <Method-Input>
      <Input-Var-Mapping FLOW-VAR="Order" Method-Var="p0">
      <Input-Var-Mapping FLOW-VAR="DeliveryTimer" Method-Var="p1">
    </Method-Input>
```

```
</Method-Input>
  <Method-Output>
    <Var-Mapping FLOW-VAR="RestaurantName" />
  </Method-Output>
</Method-Node>
<!-- Skipped the optional Certificate and Service-Exception-Handling elements -->
</Service-Node>
```

```
<!-- The Wheel Delivery service consists of two methods. Therefore, it contains a
     Method-Flow-Structure. The first method orders the food and the second one
     Charges service fee to the restaurant. -->
```

```
<Service-Node Name="Wheel Delivery" Description="Order Food and Get Payment">
```

```
  <Search-Recipe>
    <!-- Search recipe is the same as that of Check Credit node -->
  </Search-Recipe>
  <Method-Flow-Structure>
    <!-- This method orders the food from the selected restaurant. It takes four inputs: restaurant
         name, order, delivery time and address. An E-speak service method returns only one
         output. However, two output values are expected from this method: order number and
         bill amount (how much to charge the user). The Var-Mapping tags also include
         Conversion-Rule attributes which describe how to extract two expected outputs from the
         method's single output value. -->
```

```
<Method-Node Name="Order_Food" Description="Order Food">
  <Method-Name>OrderFood</Method-Name>
  <Method-Input>
    <Value>%RestaurantName</Value>
    <Value>%Order</Value>
    <Value>%DeliveryTime</Value>
    <Value>%DeliveryAddress</Value>
  </Method-Input>
  <!-- XQL is used for extracting two output values from a single method output in the following
       Var-Mapping tags. -->
  <Method-Output>
    <Var-Mapping FLOW-VAR="OrderNumber" Conversion-Rule="OrderNumber/Orderresult[0]
      Rule-Type="XQL" />
    <Var-Mapping FLOW-VAR="BillAmount" Conversion-Rule="BillAmount/Orderresult[1]
      Rule-Type="XQL" />
  </Method-Output>
```

HP docket 10008278-1
Appendix

```

</Method-Node>
<Method-Node Name="Get_Payment" Description="Get payment from the restaurant">
  <!-- This method charges a service fee to the selected restaurant. It takes two inputs:
      order number and restaurant's credit card number (or account number). It does not
      return any output since a confirmation of the restaurant's account or credit card
      is not required. -->
  <Method-Name>GetPayment</Method-Name>
  <Method-Input>
    <Value>%OrderNumber</Value>
    <Value>%RestaurantCCN</Value>
  </Method-Input>
  <!-- No method output exists for this method. Skipping Method-Output element -->
</Method-Node>
<!-- The following tags describe the arcs inside the service node's Method-Flow-Structure -->
<Arc Type="Forward" Source="Start" Destination="Order_Food" />
<Arc Type="Forward" Source="Order_Food" Destination="Get_Payment" />
<Arc Type="Forward" Source="Get_Payment" Destination="Stop" />
</Method-Flow-Structure>
<!-- Skipped the optional Certificate and Service-Exception-Handling elements -->
</Service-Node>

<!-- This service charges the bill amount, which was returned from a previous method call,
     to the user's credit card. The service consists of a single method which takes two inputs:
     bill amount and the user's credit card number. No output is required from that method. -->
<Service-Node Name="Credit Card" Description="Charge the customer credit card">
  <Search-Recipe>
    <!-- Search recipe is the same as that of Check Credit node -->
  </Search-Recipe>
  <Method-Node Name="Credit_Card_Node" Description="Charge credit card">
    <Method-Name>CreditCard</Method-Name>
    <Method-Input>
      <Value>%BillAmount</Value>
      <Value>%CustomerCCN</Value>
    </Method-Input>
    <!-- No method output exists for this method. Skipping Method-Output element -->
  </Method-Node>
  <!-- Skipped the optional Certificate and Service-Exception-Handling elements -->
</Service-Node>

<!-- The following arcs describe the service flow among the routing and service nodes of
     the composite service. Note that no start, stop and complete nodes are defined previously
     within the Service-Flow-Structure. Virtual references to Start, Stop and Complete nodes
     are used only in the arcs in order to indicate the beginning and end points of the composite
     service. Those Start, Stop and Complete nodes do not involve any method invocations
     in E-speak. Therefore, they do not have to be explicitly declared prior to their use in
     the arcs. -->
<!-- Arcs of the composite service -->
<Arc Type="Forward" Source="Start" Destination="Check Credit" />
<Arc Type="Forward" Source="Start" Destination="Restaurant Selection" />
<Arc Type="Forward" Source="Check Credit" Destination="Check Passed" />
<Arc Type="Forward" Source="Check Passed" Destination="Stop" />
<Arc Type="Forward" Source="Check Passed" Destination="And Join" />
<Arc Type="Forward" Source="Restaurant Selection" Destination="And Join" />
<Arc Type="Forward" Source="And Join" Destination="Wheel Delivery" />
<Arc Type="Forward" Source="Wheel Delivery" Destination="Credit Card" />
<Arc Type="Forward" Source="Credit Card" Destination="Complete" />

</Service-Flow-Structure>

</Meta-Model>

</Composite-Service>

```